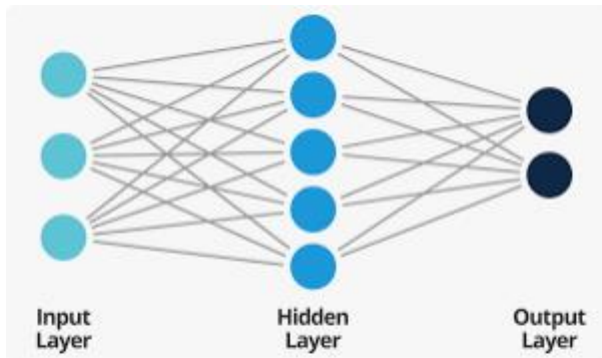
Rock Type Classification applying Neural Network (Machine Learning)

Author by: Pham Tien Duc



In this project, I made the decision to analyze the data from the drilled wells and use machine learning to forecast the facies (lithology rock type) in the oil and gas field and to make predictions for other wells based on the model. Facies is a crucial predictor of where oil and gas are located, and there are challenges when there aren't enough data from rock sample.

Using additional data and a machine learning technique is one of the best ways to interpret for Facie lithology rock type:

- **Project Overview:** One of the most crucial responsibilities for geoscientists working on development and exploratory projects is the characterization of facies. The physical, chemical, and biological conditions that a unit underwent throughout the sedimentation process are reflected in the sedimentary facies. In this project, I will analyze the data from 4 wells and the well log information to create a model that will predict the facies-lithology rock type for further wells.
- **Problem Proposition:** In this study, machine learning algorithms (Neural Networks) are trained to predict facies from well log data using data from continuous logs (NPHI, RHOB, VCL, DT & and discrete log: Facies), in order to create a model for future facies forecast for another well without facies interpretation.
- **Metrics:** As a classification strategy in this research, we employed performance metrics including recall, accuracy, and F1-Score.

- My research concludes following step:
  - Exploratory Data Analysis
  - Data processing
  - Apply the classic machine learning with ANN neural network method
  - Then, before modeling, we undertake feature engineering, scale the data, and discover and remove outliers to **enhance the performance**

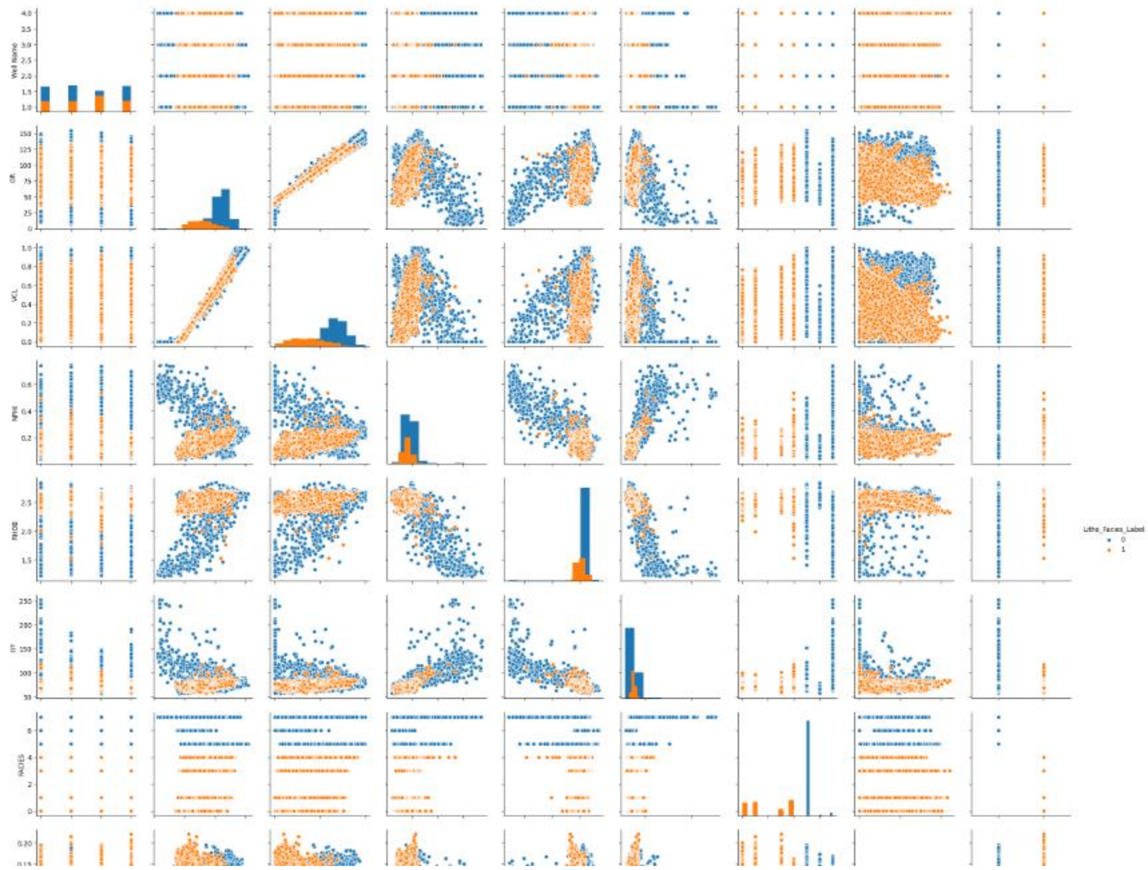
Exploratory Data Analysis (EDA)

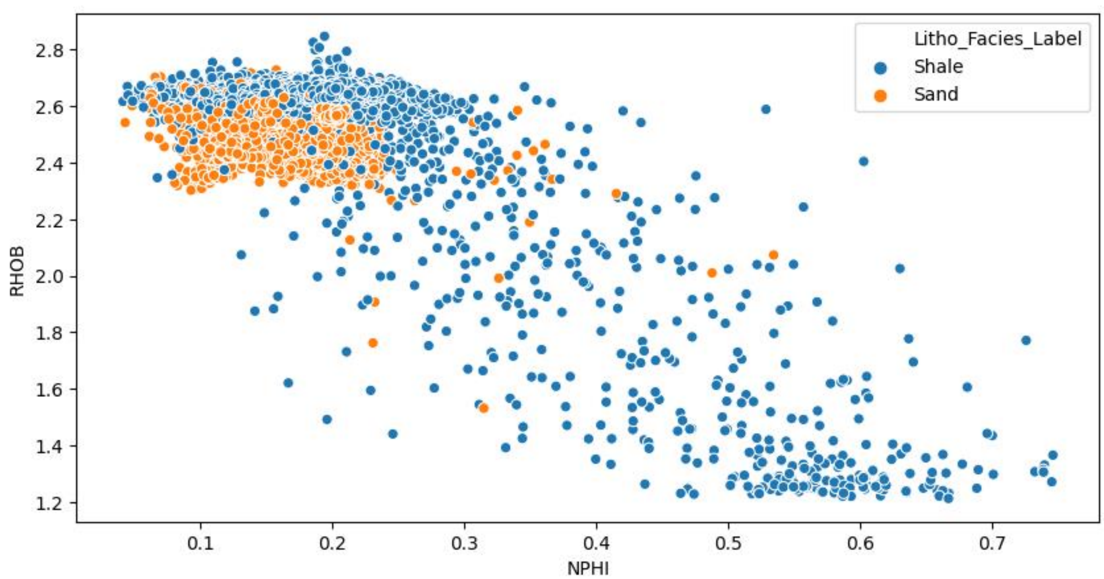
Dataset: Wells log

This well log file has over 27000 lines of data in it, including data on density porosity, bulk density, spontaneous potential, gamma rays, and resistivity. Small deep learning models can be trained and experimented with using the data.

This is done in order to anticipate the litho-facies and gain significant insights from current well data.

Data Visualization





As a result, it's critical to consider some of the following queries to provide more light on the issues:

- The relationship between the additional factor and the litho-facies?
- Which litho-facies' volume proportion is each?

Remove missing data

Remove abnormalities or characteristics about the data or input that need to be addressed have been identified.

Remove missing data

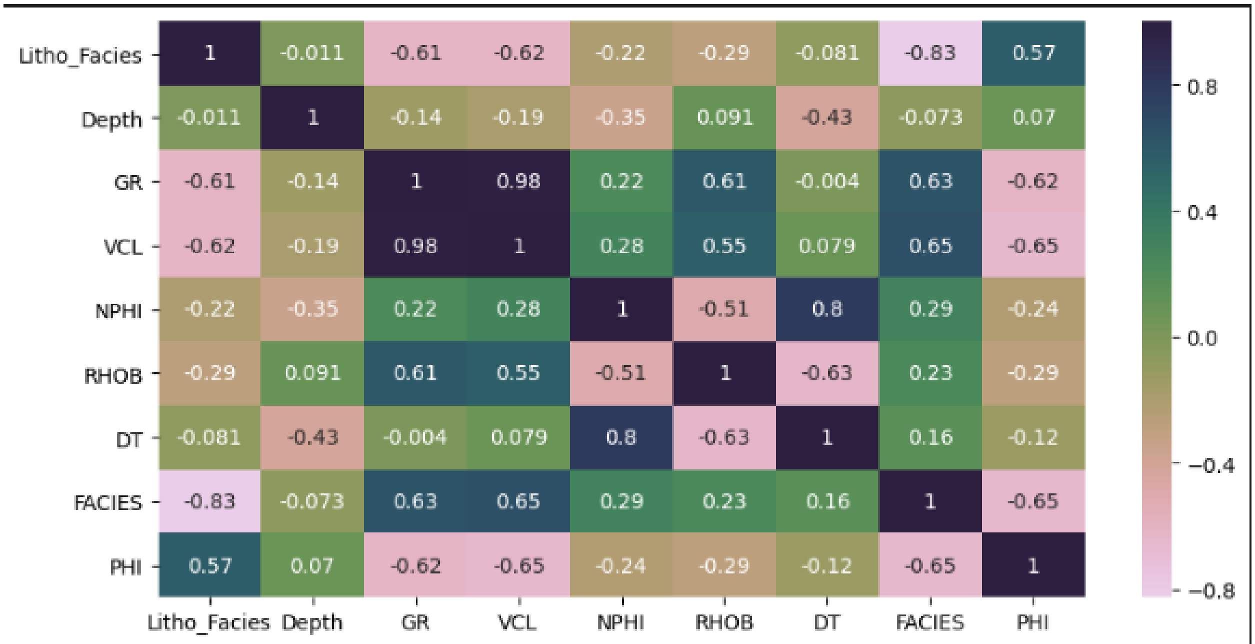
In [18]:
```
# Delete Rows with Missing Data
data = data.dropna()
# Check if any null values
data.isnull().sum()
```

Out[18]:
```
Litho_Facies          0
Formation             0
Well Name             0
GR                    0
VCL                   0
NPHI                  0
RHOB                  0
DT                    0
FACIES                0
PHI                   0
Litho_Facies_Label    0
dtype: int64
```

Question 1: The relationship between the additional factor and the litho-facies?

In order to determine which value has a better correlation with litho-facies, a scatter plot and a heatmap specifically were made to provide an answer.

This will be a helpful factor to consider when choosing a model's features in the following modeling stage.



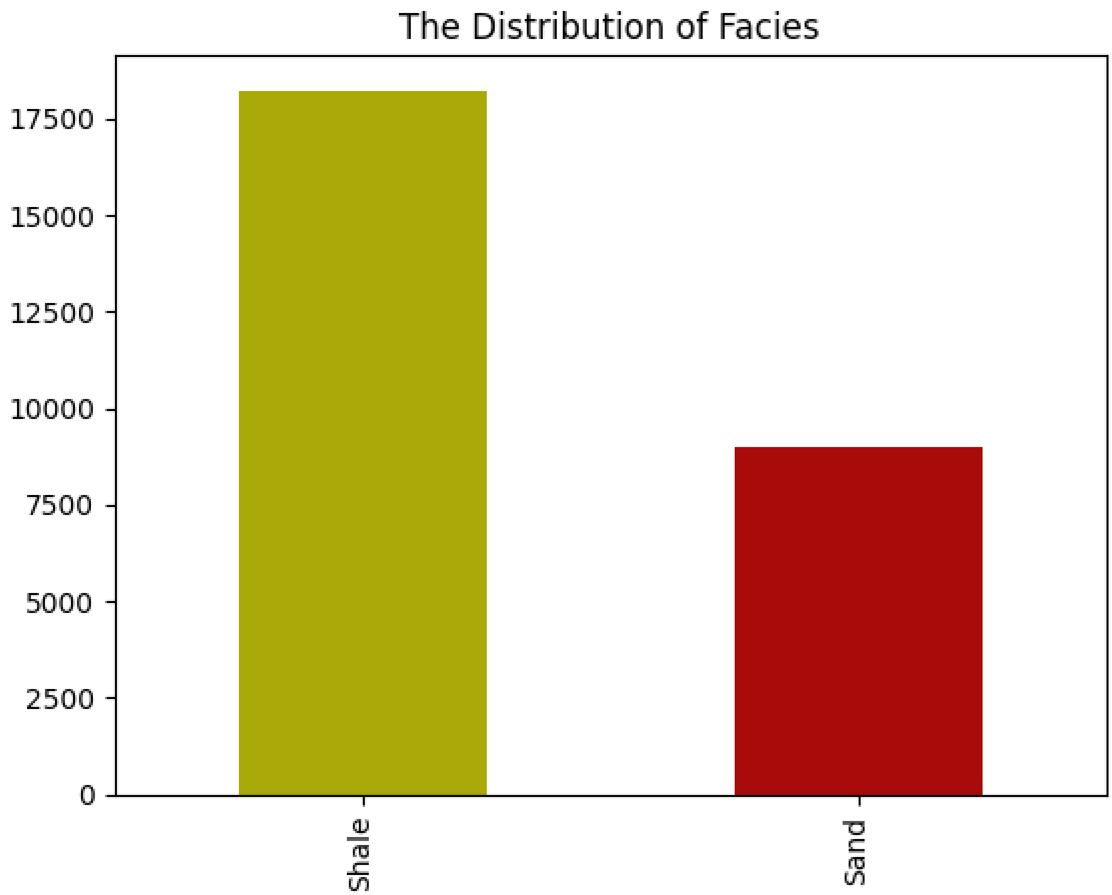
We can see the strong association between GR, VCL, PHI, and Litho-Facies.

Question 2: Which litho-facies volume proportion is each?

It will be necessary to analyze each facies' volume in order to determine which litho-facies make up the majority of our dataset and to use that information in modeling.

Since there are more chances to reserve hydrocarbons in the oil and gas industry when there is more sand present, this analyst can provide us with a clear picture of our reservoir in the field or in the most recent dataset.

The bar chart will enable you to observe the various volume fractions of sand and shale as shown below:



As can be seen, shale outperforms sand in terms of worldwide statistics.

Data Preprocessing

The data process and data engineering must be made in order to:

- Scale data by utilizing a reliable scaler, then transform
- Utilized the isolation forest technique for outlier detection and removal.

Along with the issue of having insufficient data for rock samples to feed the model (Wells formation), the data imbalance between well types (Shale and Sand) is a major issue.

There are some additional data from the [National Geological and Geophysical Data Preservation Program](https://www.usgs.gov/programs/national-geological-and-geophysical-data-preservation-program/well-log-data)

Methodology

This study employs an artificial neural network (ANN) technique for categorization. The post's conclusion provides a summary of the outcomes of this strategy.

Accuracy Evaluation in well "4"

In [19]:

```
#Using well "2" for accuracy evaluation
test_well = data[data['Well Name'] == '4'] #New data to test
data = data[data['Well Name'] != '4']
data.head(10)
```

Out[19]:

Depth	Litho_Facies	Formation	Well Name	GR	VCL	NPHI	RHOB	DT	FACIES	PHI	Litho_Facies_Label
3100.095703	0	A	1	83.580498	0.417702	0.1982	2.4784	80.315399	5	0.098686	Shale
3100.235352	0	A	1	90.570297	0.495270	0.2048	2.5065	80.236801	5	0.079254	Shale
3100.375000	0	A	1	95.361504	0.548498	0.2028	2.5277	80.528900	5	0.063111	Shale
3100.514648	0	A	1	95.203300	0.546654	0.1846	2.5138	79.506203	5	0.063171	Shale
3100.654297	0	A	1	92.073097	0.511828	0.1661	2.5249	78.292503	5	0.056550	Shale
3100.793701	0	A	1	87.148804	0.457165	0.1593	2.5481	76.769699	5	0.052213	Shale
3100.933350	0	A	1	84.778297	0.430880	0.1544	2.5719	75.580200	5	0.045494	Shale
3101.072998	0	A	1	83.430603	0.415948	0.1518	2.5788	73.989700	5	0.044303	Shale
3101.212646	0	A	1	86.701302	0.452291	0.1434	2.5733	74.265999	5	0.037151	Shale
3101.352295	0	A	1	92.956497	0.521733	0.1537	2.5624	74.970299	5	0.035490	Shale

With this as a guide the aforementioned algorithms were simple to build.

Scikit-Learn created this library. An estimator for classification in scikit-learn is a Python object that carries out the operations fit(**X_train**, **y_train**) and predicts(**X_test**).

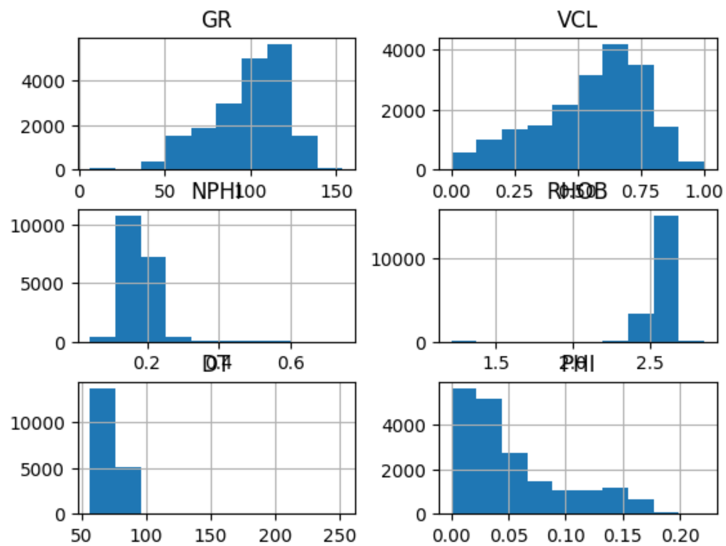
In [22]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1000)
print('Dimensions of X_train:',X_train.shape)
print('Dimensions of X_test:',X_test.shape)
```

```
Dimensions of X_train: (19034, 6)
Dimensions of X_test: (8158, 6)
```

In [23]:

```
# show distribution of training set
X_train.hist()
plt.show()
plt.tight_layout()
```



Divide the datasets in a ratio of 70:30 between training and test data.

In [22]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1000)
print('Dimensions of X_train:',X_train.shape)
print('Dimensions of X_test:',X_test.shape)
```

```
Dimensions of X_train: (19034, 6)
Dimensions of X_test: (8158, 6)
```

Outlier Detection: Isolation Forest

Training model

In [33]:

```
# Isolation Forest
iforest = IsolationForest(n_estimators=200, contamination=0.08)
# Start training the model
iforest = iforest.fit(X_train)
```

Predicting model

In [34]:

```
X_train_predict = iforest.predict(X_train)
X_train['Predict']=X_train_predict
X_train['Predict'] = X_train['Predict'].astype('category')
X_train
```

Out[34]:

	GR	VCL	NPHI	RHOB	DT	PHI	Predict
Depth							
3126.061523	0.622237	0.811949	1.215790	-0.047304	1.686263	0.365471	1
3115.172363	-1.478894	-1.533097	-1.024915	-0.393424	-1.259347	-0.655057	-1
3441.311768	-1.552908	-1.604628	0.076411	-2.092288	0.792401	2.183202	1
3305.095215	0.506583	0.626456	0.109948	0.225919	0.445369	-0.461082	1
3365.217285	-0.266803	-0.238377	-0.213697	-0.246899	-0.012051	0.252510	1
...	...	...	...	...	...	...	...
3363.113770	0.214820	0.310188	0.267208	0.180018	0.188589	0.099722	1
3526.002686	0.262138	0.316151	0.105811	0.447649	-0.409494	-0.280755	1
3740.788574	0.684511	0.722291	0.643165	0.414190	0.424973	-0.539421	1
3607.834717	0.467782	0.513927	0.406646	0.522642	0.049318	-0.319017	1
3458.075684	-0.134786	-0.310138	-1.008317	-0.281511	-0.985594	1.711454	1

19034 rows × 7 columns

In [35]:
```
X_train['y_train']=y_train
X_train = X_train[X_train['Predict'] == 1]

y_train=X_train['y_train']
X_train = X_train.drop(['Predict','y_train'], axis = 1)
X_train.head()
```

Out[35]:

Depth	GR	VCL	NPHI	RHOB	DT	PHI
3126.061523	0.622237	0.811949	1.215790	-0.047304	1.686263	0.365471
3441.311768	-1.552908	-1.604628	0.076411	-2.092288	0.792401	2.183202
3305.095215	0.506583	0.626456	0.109948	0.225919	0.445369	-0.461082
3365.217285	-0.266803	-0.238377	-0.213697	-0.246899	-0.012051	0.252510
3395.796631	0.461033	0.481916	0.219921	0.169598	0.517887	-0.303084

In [36]:
```
print(len(X_train))
print(len(y_train))
```

```
17511
17511
```

In [37]:
```
y_train.head(5)
```

Out[37]:
```
Depth
3126.061523    0
3441.311768    1
3305.095215    0
3365.217285    0
3395.796631    0
Name: y_train, dtype: int64
```

Hyperparameter tuning

The loop method, the number of interactions, and the number of layers were adjusted to reach a high level of accuracy on the training set as follows:

And from The Multi-Layer Perceptron (A Neural Network Implementation in Sklearn) library of sklearn with grid search cv to find out the best parameter apply for model

In addition, from the Sklearn Multi-Layer Perceptron (A Neural Network Implementation) package, use grid search cv to choose the optimal parameter to apply for the model.

In [38]:
```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV

parameters = {'solver': ['lbfgs'], 'max_iter': [1000,1400], 'alpha': 10.0 ** -np.arange(1, 2), 'hidden_layer_sizes':[
MLPCla = GridSearchCV(MLPClassifier(), parameters, n_jobs=-1, refit=True, verbose=3)
```

In [39]:
```
MLPCla.fit(X_train, y_train)
print(MLPCla.best_params_)
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
{'alpha': 0.1, 'hidden_layer_sizes': 20, 'max_iter': 1000, 'solver': 'lbfgs'}
```

Applying Model

In [40]:
```
MLPCla = MLPClassifier(solver='lbfgs', alpha=0.1, hidden_layer_sizes=(10,), max_iter=1000)
MLPCla.fit(X_train, y_train)
```

Out[40]:
```
MLPClassifier(alpha=0.1, hidden_layer_sizes=(10,), max_iter=1000,
              solver='lbfgs')
```

Out[50]:

	Litho_Facies	Formation	Well Name	Depth	GR	VCL	NPHI	RHOB	DT	FACIES	PHIE	Prediction
6772	1	A	4	3787.549805	61.623001	0.1983	0.1272	2.4344	73.406998	1	0.1293	1
6773	1	A	4	3787.635498	61.014099	0.1924	0.1229	2.4339	72.924004	1	0.1283	1
6774	1	A	4	3787.721680	60.157799	0.1842	0.1265	2.4399	72.998199	1	0.1282	1
6775	1	A	4	3787.807373	64.576897	0.2267	0.1229	2.4427	72.774200	1	0.1216	1
6776	1	A	4	3787.893311	64.138496	0.2225	0.1229	2.4400	72.612503	1	0.1231	1
...	...	...	...	...	...	...	...	...	...	...	...	...
6867	1	A	4	3795.709473	62.953300	0.2111	0.2028	2.5632	69.497299	0	0.1508	1
6868	1	A	4	3795.795166	62.953300	0.2111	0.2056	2.5599	69.582298	0	0.1536	1
6869	1	A	4	3795.881104	62.953300	0.2111	0.2058	2.5694	69.496300	0	0.1498	1
6870	1	A	4	3795.967041	62.953300	0.2111	0.2066	2.5688	69.559097	0	0.1500	1
6871	1	A	4	3796.052979	62.953300	0.2111	0.2033	2.5670	69.368599	0	0.1508	1

100 rows × 12 columns

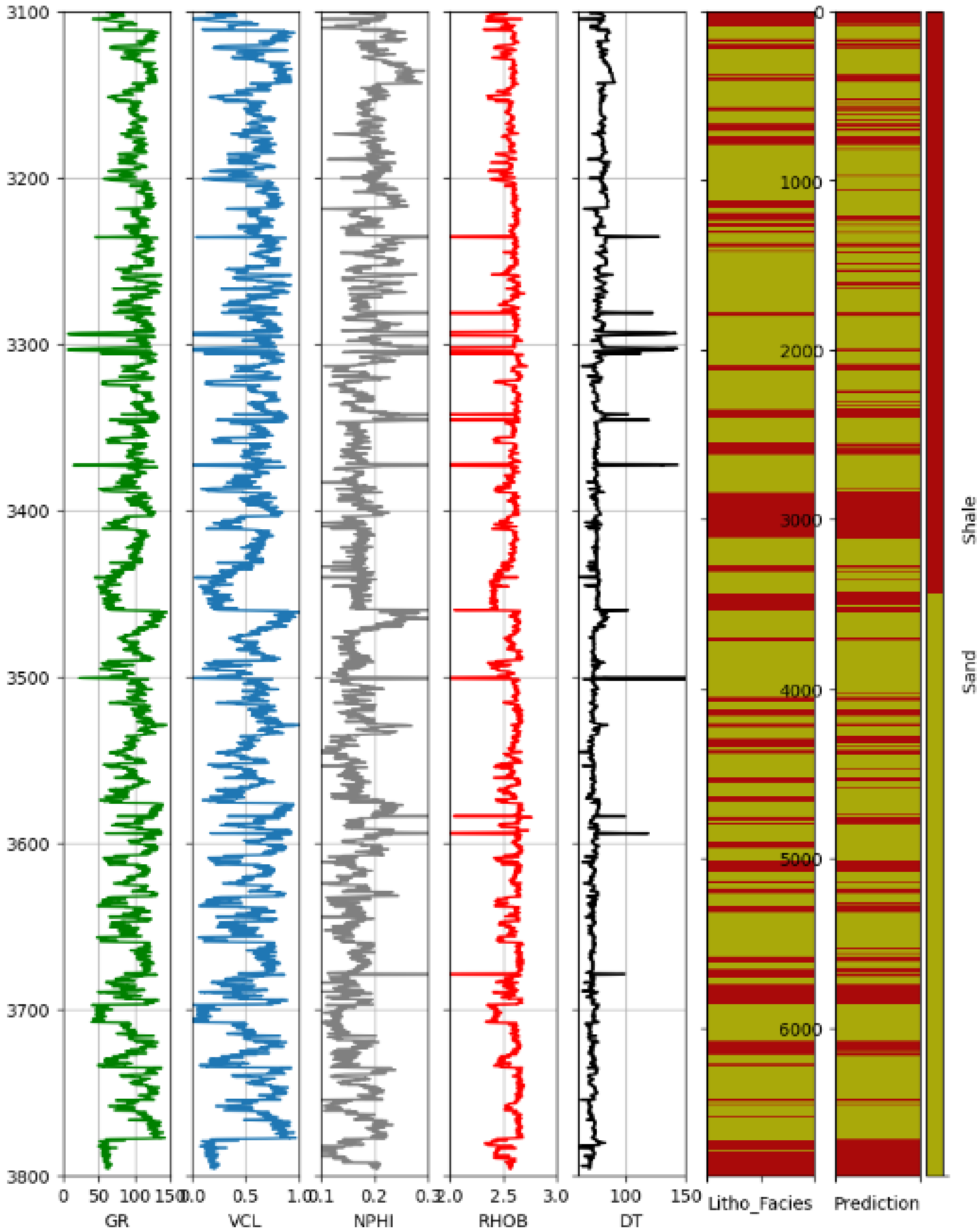
In [51]:
```
test_well.shape
```

Out[51]:
```
(6872, 12)
```

Observations

- After applying the tuning method in hyperparameters, we could improve the metrics containing accuracy and F1_score
- For more information about the comparison between Accuracy and F1_score

Result

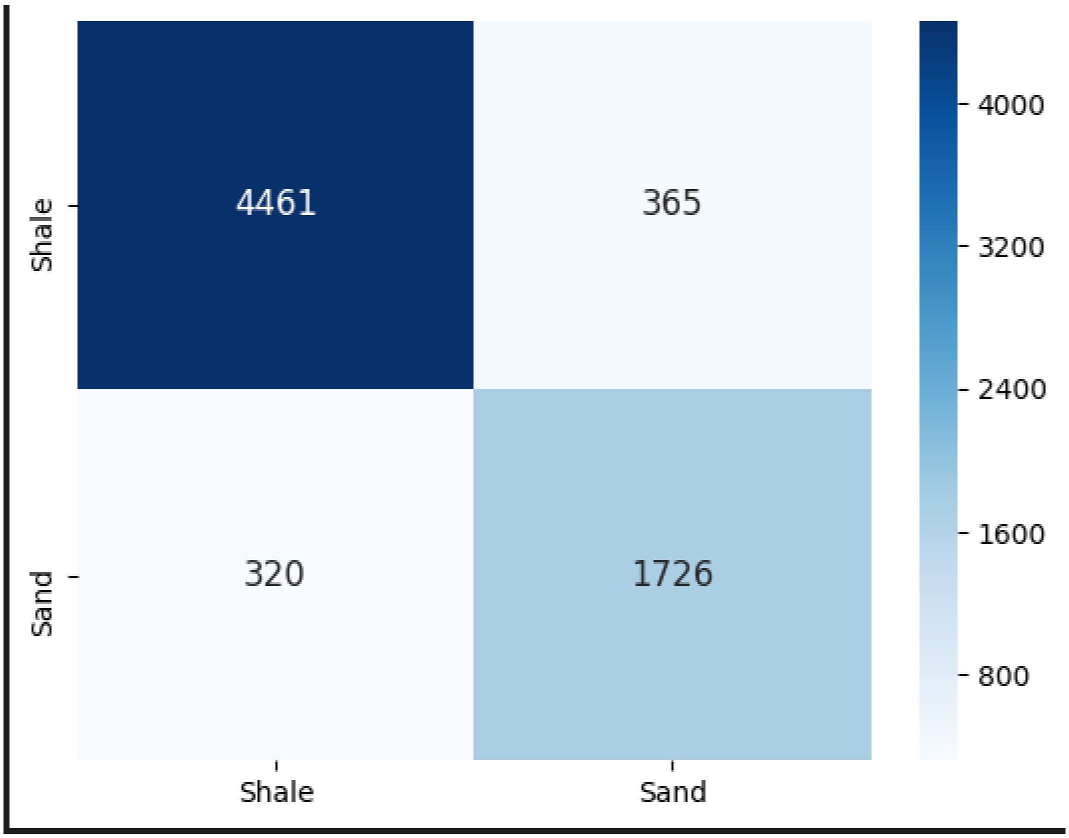


3100
3200
3300
3400
3500
3600
3700
3800
GR
VCL
NPHI
RHOB
DT
Litho_Facies
Prediction
Shale
Sand

The performance of these machine learning models is summarized using the confusion metrics, the litho-facies prediction plot in the blind well test to compare with the test data, and the plot of the density of real train/test value versus prediction.

The key performance metrics used in this study are classification metrics such as precision, recall, and F1-Score were used to validate the model, these metrics come from the concepts of True Positive, True Negative, False Positive, and False Negative.

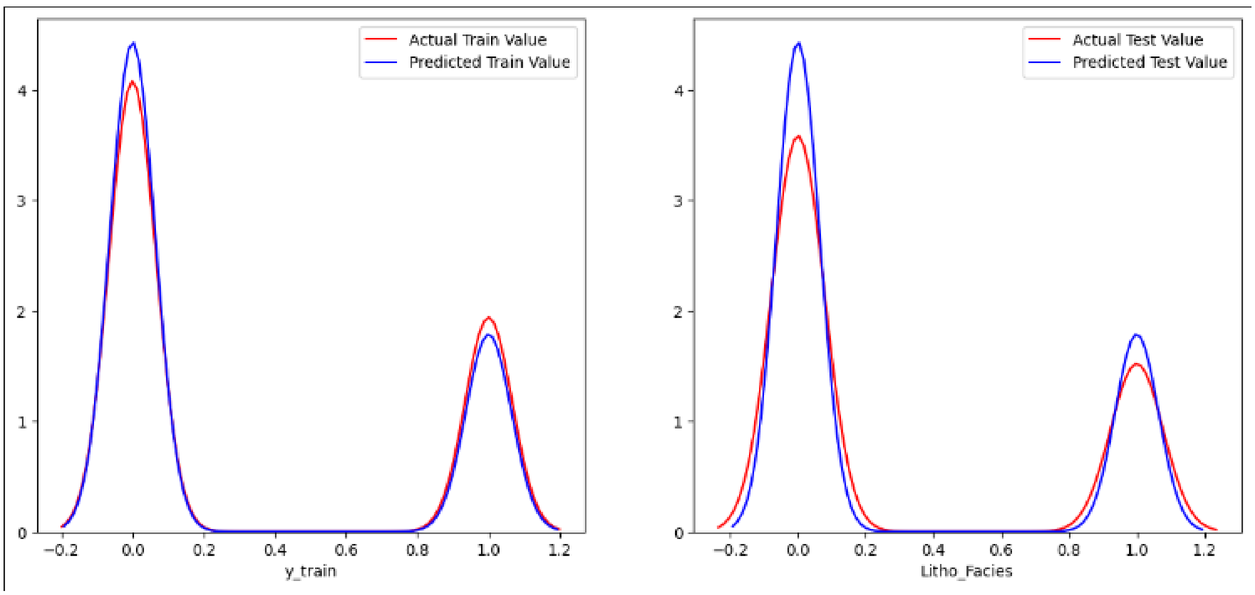
Since Shale and Sand have different distributions, the F1 score is more helpful than accuracy (particularly if your class has an uneven distribution).



	precision	recall	f1-score	support
Shale	0.93	0.92	0.93	4826
Sand	0.83	0.84	0.83	2046
accuracy			0.90	6872
macro avg	0.88	0.88	0.88	6872
weighted avg	0.90	0.90	0.90	6872

Here are the formulas for four measures that can be used to determine whether a model is appropriate or not:

Actual and predicted comparison chart



Conclusion

1. According to the blind test, with the well using the model compared with actual data in the test well, we can observe the precision is an excellent match with the real data.

2. As a result, it will be a good strategy to apply for predicting facies in other wells with a lack of data and the unbalanced data, based on the ANN model.

3. Good match based on train/test/predict values from the density visualization.

4. We may focus on two crucial areas in the following phases to enhance the outcomes: integrate more samples of data into the models, fine-tune model parameters, and use more ML approaches to compare

Improvement

In the phases that follow, we might concentrate on two critical areas to improve the results:

Adding more samples of data to the models, fine-tuning model parameters, and using more ML algorithms for comparison.

I'll give a quick summary of the attributes of our model here:

- Include more data samples in the models.
- Fine-tune model parameters
- Use more ML approaches to compare

Acknowledgements

- For more information about comparison between Accuracy and F1_score, you can follow this link (https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2)
- A Neural Network Implementation in Sklearn (https://scikit-learn.org/stable/modules/neural_networks_supervised.html#mlp-tips)
- You can find more information about Robust Scaler in this link (https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html)

Thank you for taking the time to read my article, I hope it does not waste your time.